PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 5 : G06F 12/08	A2	(11) International Publication Number: WO 93/09497 (43) International Publication Date: 13 May 1993 (13.05.93)

(21) International Application Number: PCT/US92/09417

(22) International Filing Date: 3 November 1992 (03.11.92)

(30) Priority data:
787,584 4 November 1991 (04.11.91) US

(71) Applicant: UNISYS CORPORATION [US/US]; Township Line and Union Meeting Roads, P.O. Box 500, Blue Bell, PA 19424 (US).

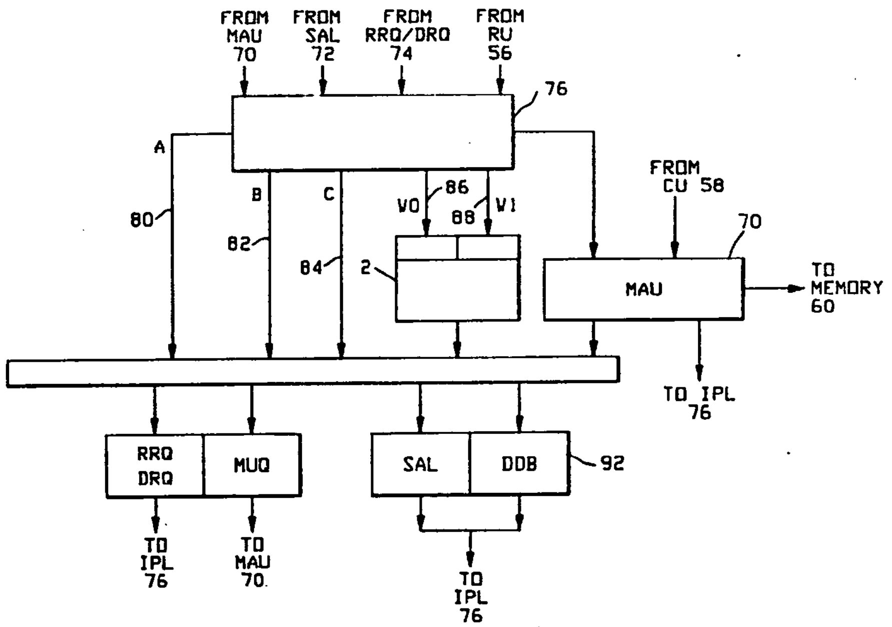
(72) Inventors: MAINO, James, G. ; 312 Dawn's Edge Lane, Exton, PA 19341 (US). NADDEO, Stanley, P. ; 101 Camsten Court, Wayne, PA 19087 (US). SNELL, Charles, K. ; 6605 Teal Loop, Frederick, MD 21701 (US).

(74) Agents: STARR, Mark, T. et al.; Unisys Corporation, Township Line and Union Meeting Roads, P.O. Box 500, Blue Bell, PA 19424 (US).

(81) Designated States: JP, KR, European patent (AT, BE, CH, DE, DK, ES, FR, GB, GR, IE, IT, LU, MC, NL, SE).

Published
Without international search report and to be republished upon receipt of that report.

(54) Title: MEMORY UNIT INCLUDING A MULTIPLE WRITE CACHE



(57) Abstract

A memory management system for a computer system controls multiple resources (80, 82, 84, 86, 88) which may be used to transfer data values between a central processing unit (56, 58), a cache memory (2) and a main memory (60). The central processing unit and the main memory generate requests for these resources which are prioritized with internal pending requests (70, 72, 74) held by the memory management system. The memory management system allocates the resources among the pending requests so that multiple requests can be processed concurrently. The cache memory (2) is organized in multiple partitions which may be written to concurrently and read from concurrently. The memory management system allocates the reading and writing of each partition of the cache memory as a separate resource.

MEMORY UNIT INCLUDING A MULTIPLE WRITE CACHE

FIELD OF THE INVENTION

This invention relates generally to memory management apparatus for use in a digital data processor and particularly to techniques for improving the efficiency of a memory system including the retrieval and storage of data in a cache memory.

BACKGROUND OF THE INVENTION

A memory management system controls the access of multiple requestors in a digital data processing system to data and program code in a main memory. In many computer systems, all memory access requests which either fetch or store data are made through a cache memory. A cache memory is a memory that is packaged as an integral component of a processing unit in the computer system. The cache is much smaller than main memory. Its purpose is to emulate the main memory, but with a much faster access time. Since a cache is smaller than main memory, it can only contain a subset of the contents of the main memory at any one time. A directory is used in a cache to identify which parts of the main memory are resident in the cache.

The prior art contains many techniques which improve the ability of cache memory systems to furnish data to and receive data from the central processing unit. U.S. Patent 4,208,716 to Porter, et al. discusses a cache which is partitioned into multiple levels. While a write request is being performed on one level of the cache, one or more read operations from other cache levels may be performed

U.S. Patent 4,905,141 to Brenza discusses the use of a cache which is divided into partitions, where a plurality of processors may each access any of the partitions. Each processor has a partition look-aside table (PLAT) which performs address translation, identifying the partitions and congruence class identifiers of the cache entries most recently referenced by that processor. The cache is capable of performing one store or fetch operation per partition per clock cycle for data which are in the PLAT. Only a subset of the data in cache are listed in the PLAT. If a reference is made to a data value which is not in the PLAT, a global request is issued to all of the partitions. This global request requires all cache partitions for one clock cycle.

U.S. Patent 4,794,521 to Ziegler, et al. discusses a cache capable of concurrently working on the completion of multiple cache accesses. The cache is divided into sections. Each section can handle one access per clock. If the access causes a cache miss, then the cache has the ability to put the first access in a pending-access-completion state while the memory access occurs, and of accepting another cache access. Each section can handle up to three such pending accesses.

SUMMARY OF THE INVENTION

Previously, cache memory systems did not support multiple writes to cache and reads from cache in a single clock cycle. The present invention is embodied in an improved computer system with cache memory, in which requests from multiple sources, including multiple cache write requests and at least one cache read, may be accomplished in a single clock cycle. Those requests which can be concurrently serviced are automatically funnelled to the appropriate cache or memory resources.

According to one aspect of the invention, the cache is divided into two logically distinct page groups. Each page group comprises a plurality of cache pages. Each page group is provided a unique write address and write data signal. Each cache write request consists of data and the cache address and page number in which the data are to be stored. When two cache requests try to write to the cache simultaneously, both write operations are successful so long as they are performed on different page groups.

In order to efficiently provide cache read and write requests from the central processing module (CPM), a Memory Unit (MU) is provided within the CPM. The Memory Unit is responsible for managing the data cache.

According to another aspect of the invention, the MU simultaneously receives memory access requests from multiple requestors within the central processing module. The MU allocates several resources including the two cache memory write paths and at least one cache memory read path. In order to achieve a high level of efficiency, the MU determines what resources are requested by each of its pending requests and allocates these resources so that multiple requests may be handled simultaneously.

BRIEF DESCRIPTION OF THE DRAWINGS

FIGURE 1 is a block diagram of a cache system which supports two simultaneous write operations.

FIGURE 2 is a block diagram of a central processing module which controls the cache system shown in FIGURE 1.

FIGURE 3 is a block diagram of a Memory Unit which controls the cache shown in FIGURE 1.

DETAILED DESCRIPTION

The following is a description of an exemplary memory management system in accordance with the present invention.

FIGURE 1 shows a cache memory system in accordance with the present invention. Cache memory 2 is a set associative, store-in cache capable of storing up to 4096 data words. Cache memory 2 is partitioned into four direct mapped cache pages 10, 12, 14 and 16, each having a 1024 word capacity. For each 1024 word cache page, 10, 12, 14 and 16, there is a corresponding address array 4, which performs address translation to determine whether a particular data value is stored locally in the cache.

A direct mapped cache is one in which the data associated with any particular absolute address can only be stored in a fixed location within the cache. The ten least significant bits of the absolute address of the data value are used to determine the location in a cache page where the data value is stored. The third through the tenth bits of the absolute address define the congruence class for a data value.

The congruence class addresses four words, having consecutive address values, which are fetched from main memory into the cache in order to take advantage of spatial locality of reference. Spatial locality of reference is the observed tendency of programs to refer to data fetched from memory which have address values that are close together. The group of words in response to a request for a single memory word is often referred to as a line of memory words. The two least significant bits, of the ten-bit address value, which are defined as the word number, define the position of a particular data word within the line of words indicated by the congruence class address.

Each address array 4 contains 256 entries, each of which points to a separate four word line in the cache, one line for each congruence class. To determine whether a data value is stored locally in cache, the entry in the address array 4 corresponding to the congruence class of the data value is accessed. The address array entry holds the 19 most significant bits (called the page select field or PSF) of the absolute address of the data value stored in that portion of the cache. If the page select field in the selected entry from one of the four address arrays matches the 19 most significant bits of the address of the requested data then the requested data value may be in the cache memory 2. The address array entry also includes flags which indicate whether the data cache entry is valid, and whether it has been modified since it was first brought into the cache 2 from main memory (not shown). If a valid entry exists in the address array 4 for the desired congruence class and page select field, then the cache page 10, 12, 14 or 16, corresponding to the address array 4 which contains the matching PSF entry, contains the desired data value.

When a data value is stored in the cache 2, the congruence class and the word number of the absolute address of the data value determines the location within one of the cache pages 10, 12, 14 and 16 in which the data value is to be stored. This storage operation typically requires replacement of data in one of the pages in the cache 2. To select which of the four pages 10, 12, 14 or 16 is used, a Least Recently Used (LRU) algorithm is employed. The LRU algorithm takes advantage of temporal locality of reference which is a recognized tendency of programs to reference the same data values in a relatively short time interval.

The line of data values in the page 10, 12, 14 or 16 which was accessed least recently is selected to be replaced by a four-word line containing the requested data value. Providing four pages 10, 12, 14 and 16 allows the cache 2 to store up to four data words whose absolute addresses map to the same cache address, before it is necessary to castout any data words. This improves the cache hit percentage, which is the ratio of satisfied requests for data to the total number of requests.

In the present invention, the cache pages 10, 12, 14, 16 are divided into two groups 20 and 22, each consisting of two pages 10 and 12 in group 20 and 14 and 16 in group 22. There are two write address registers 24 and 26; two write data registers 30 and

32; and four write enable registers 40, 42, 44 and 46. Data values are written to page 10, 12, 14 or 16 when respective write enable register 40, 42, 44 or 46 is set. When the system writes to either page 10 or page 12, the data provided in write data register 30 is written to the cache at an address provided in write address register 24. Similarly, when the system writes to either page 14 or page 16, the data provided in write data register 32 is written to the cache at an address provided in write address register 26.

When two simultaneous requests to write data into the cache occur, performance is enhanced by servicing both requests in a single clock period, so long as the two requests are writing to different page groups 20 and 22. If two write requests for the same cache page group occur at the same time, one of the requests is queued, and two clock cycles are required to service both requests.

It is understood by one skilled in the art that a larger number of write address registers 24, 26 and write data registers 30, 32 registers can be provided. In particular, the number of pages in the cache memory could be increased and the write address registers and write data registers could be set equal to the number of cache pages. In this alternative embodiment, simultaneous cache write operations would be possible within a single clock cycle, so long as the data values are written to respectively different pages. This variation can improve cache performance over the exemplary system for simultaneous write requests to pages which would, in the exemplary embodiment, be in the same page group. It may, however, require more complex logic to detect and resolve simultaneous write requests for the same page than is required when the exemplary page groups are used.

In the present embodiment of the invention, a single read address value is applied to all of the cache partitions from input priority logic of a memory unit which contains the cache memory 2. The target data value is then selected from among the values provided by the various partitions. The memory unit and the input priority logic are described below in reference to FIGURE 3. Both the read operation and two write operations may be performed in a single period of the system clock signal. In the exemplary embodiment of the invention, the read operation is performed at a time later in the clock cycle than the parallel write operations.

It is contemplated that the exemplary cache memory system may be extended to allow multiple simultaneous read operations. This extension could take place without significant change to the cache memory. In one contemplated implementation, the input priority logic of the memory unit would be modified to provide a separate read address value to each partition group or to each individual partition.

The cache memory system 2 is a part of the data processing system which includes an instruction execution unit and a main memory. In order to properly describe the advantages gained by this cache memory architecture, it is helpful to place it in its intended environment.

FIGURE 2 shows a block diagram of a Central Processing Module (CPM) 50 used in systems such as the Unisys A19 processing system. The CPM 50 comprises four primary subsystems: the Memory Unit (MU) 52, including the cache memory 2, the Execution Unit (EU)

54, the Reference Unit (RU) 56, and the Code Unit (CU) 58. Also included in the CPM 50 is a clock signal generator 51 which provides a periodic clock signal, CK, to each of the units 52, 54, 56 and 58.

The MU 52 is the connection between all CPM 50 subsystems and main memory 60. The MU 52 is responsible for managing all memory traffic, as well as performing all data fetch and store operations requested by the Reference Unit 56. The MU 52 manages the cache memory system 2 and the other resources, described below, which are available for accessing stored data values in order to decrease the effective memory access time.

Reference Unit 56 provides jobs to MU 52, which are requests for the MU to fetch data. In addition, the RU 56 provides the address values for store requests which will be processed as the data values to be stored are provided by the EU 54. Since the cache 2 is a store-in cache memory, the MU 52 fetches the data values to be overwritten from main memory as soon as the address values are provided by the RU 56. Thus, when the EU 54 provides the data value, it may be stored locally in the cache memory 2 and only returned to the main memory 60 when it needs to be castout.

In general, data values are transferred to or from the MU 52 through the Distributed Data Buffer (DDB) 92. Data may be written to and read from the DDB 92 by the MU 52, EU 54 and RU 56. This buffer is a general purpose register file which is duplicated in the EU 54, RU 52 and MU 52. In the exemplary embodiment, the RU 52 instructs the MU 52 to use the DDB to transfer the data value to or from the main memory 60 through the cache memory 2.

The Code Unit 58 provides jobs to MU 52, which are requests to fetch code from main memory 60. As set forth below, these requests are handled by the MU 52 at a lower priority than data requests.

The Execution Unit 54 evaluates instructions provided by the CU 58 using data values provided by the MU 54 in response to fetch requests provided by the RU 52. The data produced by evaluating the instructions is provided to the MU 54, via the DDB 92, to be stored in the cache memory 2.

FIGURE 3 is a block diagram of a Memory Unit 52. MU 52 includes six resources which are used to effect data transfer among the address arrays 4, the DDB 92, the data cache pages 10, 12, 14 and 16, and main memory 60. These resources are the address array path (A-path) 80, the DDB path (B-path) 82, the cache read path (C-path) 84, the two cache write paths 86, 88 (W0 and W1), and the MAU request path (M-path) 90. Input Priority Logic (IPL) 76 accepts input signals from several functions (or requestors) which request cache and main memory access. Each of these requestors uses a different combination of the resources of the MU 52.

The IPL 76 serves as the main job issuing control element for the MU 52. None of the requestors individually requires all of the resources of the MU 52. Accordingly, the IPL 76 allocates as many of the resources as possible to service multiple requestors within a single period of the system clock signal CK. When, due to resource conflicts, the IPL 76 cannot service all of the requestors in a single clock period, the unprocessed jobs are held in a queue in the IPL along with their accompanying data. The IPL 76 attempts to reissue these jobs during successive clock cycles until they are completed.

IPL 76 receives input signals requesting resources from five different requestors. These requests, also known as jobs, are prioritized by the IPL. The requestors (in order of priority) are the Memory Access Unit (MAU) 70, the Store Address List (SAL) 72, the Return and Replace Queue (RRQ) / Deferred Reference Queue (DRQ) 74, and the Reference Unit (RU) 56. The SAL 72 and the RRQ/DRQ 74 are functions within the MU 52. The remaining input signals are received by IPL 76 from functions external to the MU.

The order of priority is designed to ensure integrity of the cache 2 operation. Because a store-in cache is used, jobs from MAU 70 which move data from main memory 60 to cache 2 must occur before jobs from the SAL 72 which store data into the cache 2.

For example, a store operation may occur as follows: when the CU 58 encounters a store instruction it passes the instruction to the RU 56 which, translates the data address in the instruction into a memory address. The RU 56 then requests a memory read operation of the MU 52 to localize the data in the cache 2 and, at the same time, places an entry for the store operation into the SAL 72 of the MU 52. When the target memory address is localized in the cache memory and the data value to be stored in that address, is marked as a valid entry in a DDB register, the entry in the SAL 72 becomes active as a request to store the data value in the designated DDB register into the cache memory 2.

As illustrated above, the reference unit may process each instruction provided by the code unit to generate multiple memory access requests. Memory access requests generated from several instructions may be pending in the IPL 76 at any given time. Any pending memory access requests which can be handled concurrently by the MU 52 are issued concurrently by the IPL 76.

If, at the time the value in the DDB register becomes valid, the target memory address is not localized in the cache due to an intervening castout operation, the MU issues a request to read the data from main memory 60 and holds the write request in the SAL 72 as an inactive entry. The IPL 76 always processes the oldest active SAL entry.

The RRQ 74a and the DRQ 74b share a single input port to provide signals to the IPL 76. There is logic within RRQ/DRQ 74 to ensure that RRQ jobs always receive priority over DRQ jobs. Jobs are placed in the Deferred Reference Queue 74b when an attempt is made to fetch data before it is stored in the cache. Jobs are placed in the Return and Replace Queue 74a when data must be written to or fetched from main memory 60, but the resources are not available. In the event that the same data value in main memory 60 is being requested by an RRQ 74a job and a DRQ 74b job, it is necessary to continue delaying the DRQ 74b job (e.g. a transfer from cache 2 to the DDB 92) until satisfaction of the RRQ 74a job (e.g. a transfer from main memory 60 to cache 2).

The RRQ/DRQ 74 jobs are processed before the current jobs from RU 56. This ensures the correct ordering of jobs because the RRQ/DRQ jobs were issued earlier and then delayed by being placed in the queue.

For each clock cycle, the IPL 76 attempts to service an MAU 70 job first. With the remaining resources, the IPL 76 then attempts to service a SAL 72 job. If there are still unallocated resources, the IPL attempts to service the RRQ/DRQ 74, and if some of the resources are still available, it attempts to service a job from the RU 56. In addition to jobs serviced by the IPL, an input signal from the Code Unit 58 is provided to the MAU 70 to fetch code from the main memory 60 via MAU access path 90. Requests from the CU 58 always have the lowest priority, so they are not serviced until all of the jobs from the other requestors (controlled via IPL 76) which require the MAU access path 90 are complete. Requests from the CU 58 are processed last, because they bring additional instructions to the CPU, and, so, are likely to generate even more memory requests.

As described above, the MU 52 attempts to process several pending requests at the same time by allocating its available resources among the requests. TABLE 1 illustrates the resources that may be required to complete jobs provided by each of the requestors. The available resources include: an address array lookup and update path (A), a distributed data buffer (DDB) write path (B), a data cache read path (C), an MAU request path (M) and a data cache write path (W). In table 1, the term "Undef." means that the class value is not defined for the requestor.

TABLE 1

Requestor Path Requirements

Class	0	1	2	3
MAU	A	W	W,B	A,B
SAL	A	M	W	
Undef.				
RRQ/DRQ	A,B,C	Undef.	M	M,C
RU	A,B,C	B	A,B	Undef

Note:
CU requests are not provided to the IPL, but are provided directly to the MAU. CU requests are only serviced when no IPL requestor is using the M-path to access main memory.

A request consists of a request valid bit, a two-bit request class code, and all data needed to complete the requested transaction. When the IPL 76 allocates a resource to a job of one requestor, the IPL provides the valid bit, the class code, and the data in the request to the allocated resource. As shown in the TABLE, a single requestor may issue up to four different classes of requests, each with different resource requirements. There is no constraint on the class codes of the requests submitted by multiple requestors in a single CPU clock cycle, but any conflict in resource requirements among pending jobs results in a delay, through the IPL, for one or more of the jobs.

The following is an example of the operation of the system. At the start of a clock cycle, the IPL 76 services a class 1 job from MAU 70, only requiring the use of the first data cache write path, W0 86. The IPL 76 has resources available to service a class 2 SAL 72 job which only uses the second data cache write path, W1 88. Even with these two jobs, There are still resources available to service a class 2 job from RRQ 74a. This job only uses the MAU request path 90. Finally, If the next pending job from the RU is a class 1 job, which requires the DDB write path 82, then all four requests can be serviced in a single clock cycle.

The above example illustrates two features of the invention. First, the system can service two separate cache write jobs, one each from the MAU 70 and the SAL 72, in a single clock cycle, although, in the disclosed embodiment, these write jobs are required to be in respectively different page groups. The IPL 76 uses the dual write cache capability shown as shown in FIGURE 1, and previously discussed. Second, the system is capable of servicing up to four different jobs in a single clock cycle. As shown in TABLE 1, several combinations of jobs can result in four simultaneous job executions in a single clock cycle. These combinations may include two writes and two reads, or one write and three reads (one of which is a read from main memory 60, as opposed to cache 2).

An additional feature of the present invention is the cache fill operation. This operation allows further improvement in the utilization of resources. As shown in TABLE 1, an RU 56 request ordinarily requires the DDB write path 82, in addition, and the A-path 80 and C-path 84 may be required. If, during a given clock cycle, a request from RU 56 cannot be processed because a higher priority requestor is using the B-path 82 or the C-path 84, but the A-path 80 is still available, a cache fill job is performed.

For this job, the RU 56 request is permitted to perform an address array lookup to determine cache locality for the request. If a cache hit occurs on the cache fill job, no action is taken and the RU request is reissued from the IPL during the following clock cycle. If, however, a cache miss occurs, then a request is issued to the MU MAU Request Queue (MUQ) 96 which stores the request as an MU job, requesting transfer of data from main memory 60 to cache 2. The cache fill job allows the request for data from main memory 60 to be issued at least one clock cycle earlier than would be possible if the RU 56 job were to wait for the B-path 82 and/or C-path 84 to become available.

The following pseudo-code program segment illustrates the job allocation algorithm used by the IPL 76.

```
IF MAU Request is valid THEN
     Issue the MAU Request
     Remove MAU Resources from available resources list
     Accept new MAU Request
ELSE
     Accept new MAU Request
IF SAL Request is valid THEN
     IF all SAL requested resources are available
          Issue the SAL Request
          Remove SAL resources from avail. resource
list
          Accept new SAL Request
     ELSE
          HOLD SAL Request
ELSE
     Accept new SAL Request
IF RRQ/DRQ Request is valid THEN
     IF all RRQ/DRQ requested resources are available
          Issue the RRQ/DRQ Request
          Remove RRQ/DRQ resource from avail. list
          Accept new RRQ/DRQ Request
     ELSE
          HOLD RRQ/DRQ Request
ELSE
     Accept new RRQ/DRQ Request
IF RU Request is valid THEN
     IF all RU requested resources are available
          Issue the RU Request
          Remove RU resources from avail. resource list
          Accept new RU Request
          ELSE
               IF RU Request Needs A-path
                 and A-path is available THEN
                    Attempt Cache Fill operation
               ELSE
                    HOLD RU Request
     ELSE
          Accept new RU Request
```

This program segment illustrates the operation of the input priority logic (IPL) 76. For the sake of clarity, the process is presented in the form of a program, in the exemplary embodiment of the invention, however, the algorithm is implemented as a finite-state machine using logic circuits. One of ordinary skill in the art of logic circuit design could readily implement suitable circuitry for the IPL 76 from the program segment and the description of the IPL 76 circuitry given in the specification.

The IPL 76 accepts the jobs to be executed during one cycle of the system clock signal and then selects and issues the jobs during the next clock cycle. From the above program segment, it can be seen that jobs from the MAU 70 are always executed. Jobs originating from the SAL 72, RRQ/DRQ 74, and RU 56 are executed only in the stated priority order, and only if there is no conflict with a higher priority job. Jobs which cannot be completed because the necessary resources are unavailable are held in the IPL 76 and the IPL attempts to process them during the next successive clock cycles.

It is understood by one skilled in the art that many variations of the embodiments described herein are contemplated. These include different cache sizes, different number of cache pages, different page group size, and different number of page groups as well as different number of requestors and different number of available resources. While the invention has been described in terms of an exemplary embodiment, it is contemplated that it may be practiced as outlined above with modifications within the spirit and scope of the appended claims.

The invention claimed is:

1. Apparatus for managing access to data values in a cache memory which is divided into a plurality of partitions, wherein each of the data values has a respective address value, the apparatus comprising:

means for accessing each cache memory partition separately for cache write operations;

means for mapping each data value to be accessed into a respective one of the plurality of cache memory partitions based on the address value of the data value; and

means, coupled to said accessing means, for concurrently a) storing ones of said data values in selected ones of said cache memory partitions, wherein the stored data values have respectively different partition mappings in said cache memory, and b) fetching at least one of said data values from one of the selected cache memory partitions.

2. Apparatus according to claim 1, wherein the system further includes means for generating a periodic clock signal and means for writing data to each cache memory partition and for reading data from at least one cache memory partition during one period of the clock signal.

3. Apparatus for managing access to data values in a cache memory which is divided into a plurality of partitions, where each data value has a corresponding address value, the apparatus comprising:

means for associating ones of said plurality of cache memory partitions into a plurality of cache memory partition groups wherein each partition group includes at least two cache memory partitions;

means, responsive to the address values of respective first and second ones of said data values, for selecting respective first and second cache memory partition groups into which the first and second data values are to be stored;

means for translating the first and second address values of the respective first and second data values into first and second cache memory address values within the respective first and second selected cache memory partition groups;

means for providing said first and second cache memory address values and said first and second data values to each of the cache memory partition within the respective first and second cache memory partition groups;

means for enabling the storage of said first and second data values using the respective first and second cache memory address values within the selected first and second cache memory partitions; and

means for concurrently storing the first and second data values into the respective first and second selected cache memory partitions.

4. Apparatus according to claim 3, further comprising:

means for providing a periodic clock signal; and

means for writing data into each cache memory partition group and for reading data from each cache memory partition group during a single period of the clock signal.

5. A memory management system which controls access to data in computer system that includes an instruction processing unit, a plurality of registers, a cache memory divided into a plurality of partitions, and a main memory, the system comprising:

a plurality of queueing means for storing memory access requests provided by the instruction processing unit, the main memory and the memory management system;

a plurality of memory unit resource means for transferring data values among the instruction processing unit, the plurality of registers, the cache memory and the main memory;

logic means, coupled to the plurality of queueing means, for a) receiving memory access requests, b) determining which of the plurality of memory unit resource means are required to satisfy each of the requests, and c) assigning one or more of the memory unit resource means to selected ones of the requests which can be processed concurrently; and

means for concurrently processing the selected requests.

6. A memory management system according to claim 5, wherein:

the cache memory includes a plurality of partitions; and

the memory management system further includes means for assigning each of said cache memory partitions as a separate resource and means for simultaneously storing respective data values into each of said cache memory partitions.

7. A memory management system according to claim 5, further comprising means for concurrently a)

storing data into each of said cache memory partitions and b) reading data from at least one of said cache memory partitions.

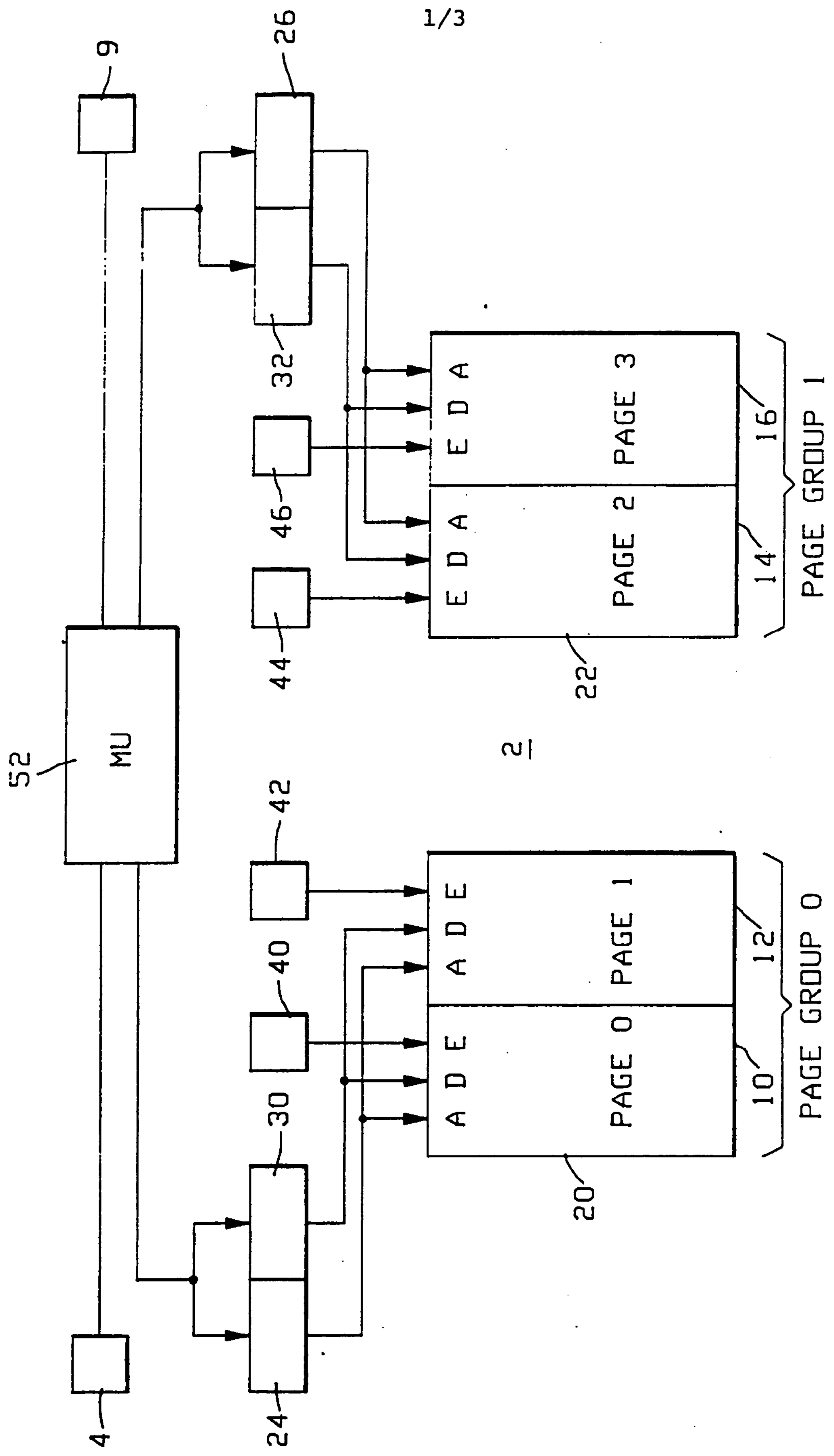
8. A memory management system according to claim 5, wherein the queueing means comprises:

store address list means for maintaining an ordered list of all pending requests to store data from the central processing unit into the cache memory;

deferred reference queue means for maintaining an ordered list of all pending requests to reference data in the cache memory; and

return and replace queue means for maintaining an ordered list of requests to transfer data from the main memory to the cache memory and from the cache memory to main memory.

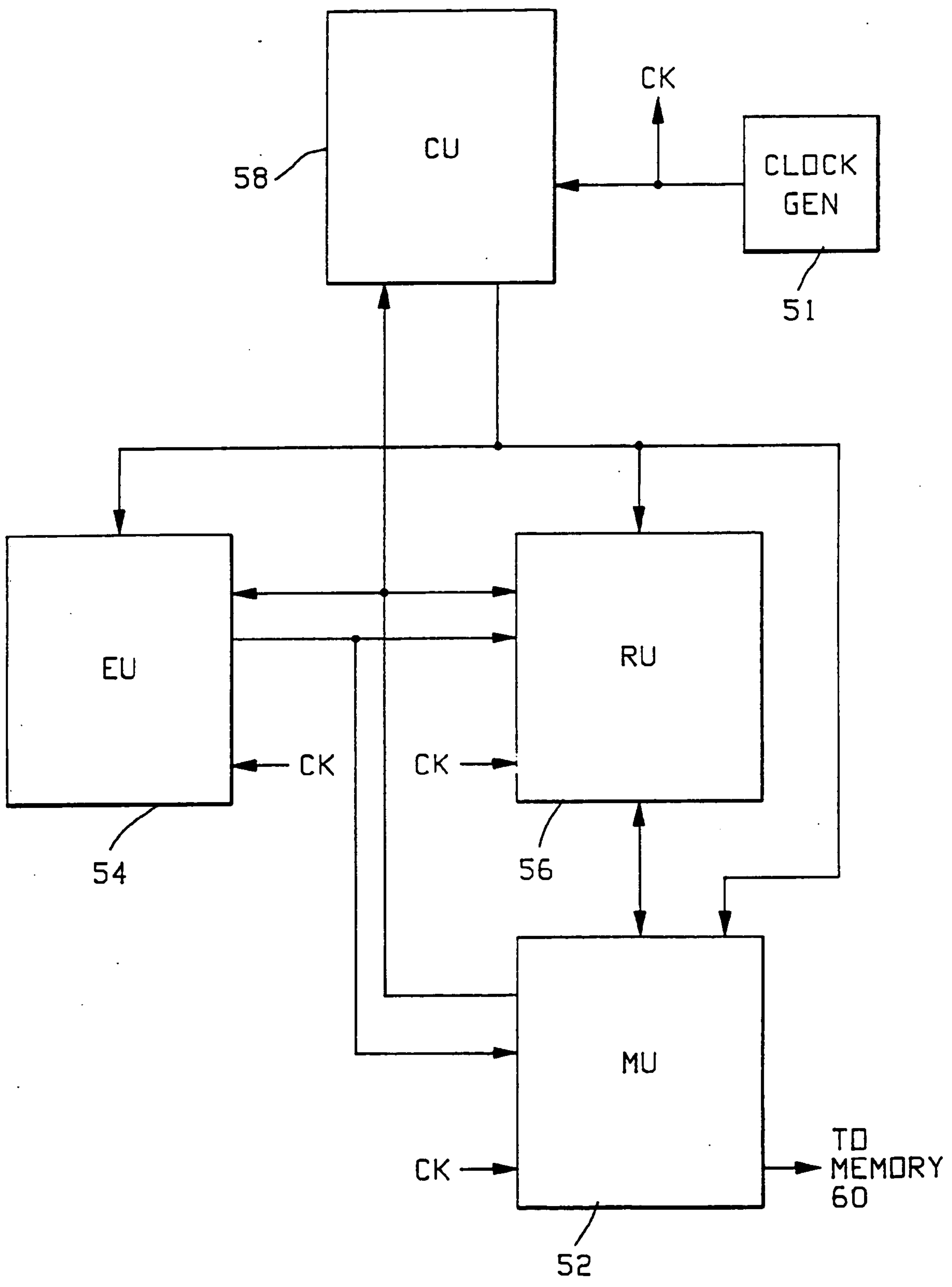
9. A memory management system according to claim 5, wherein the instruction processing unit includes means for generating multiple memory access requests from each processed instruction and the logic means includes means for concurrently processing memory access requests generated from respectively different instructions.



4
9
24
26
30
32
40
42
44
46
52
MU
2
A D E
A D E
E D A
E D A
PAGE 0
PAGE 1
PAGE 2
PAGE 3
10
12
14
16
20
22
PAGE GROUP 0
PAGE GROUP 1

FIG.1

FIG. 2



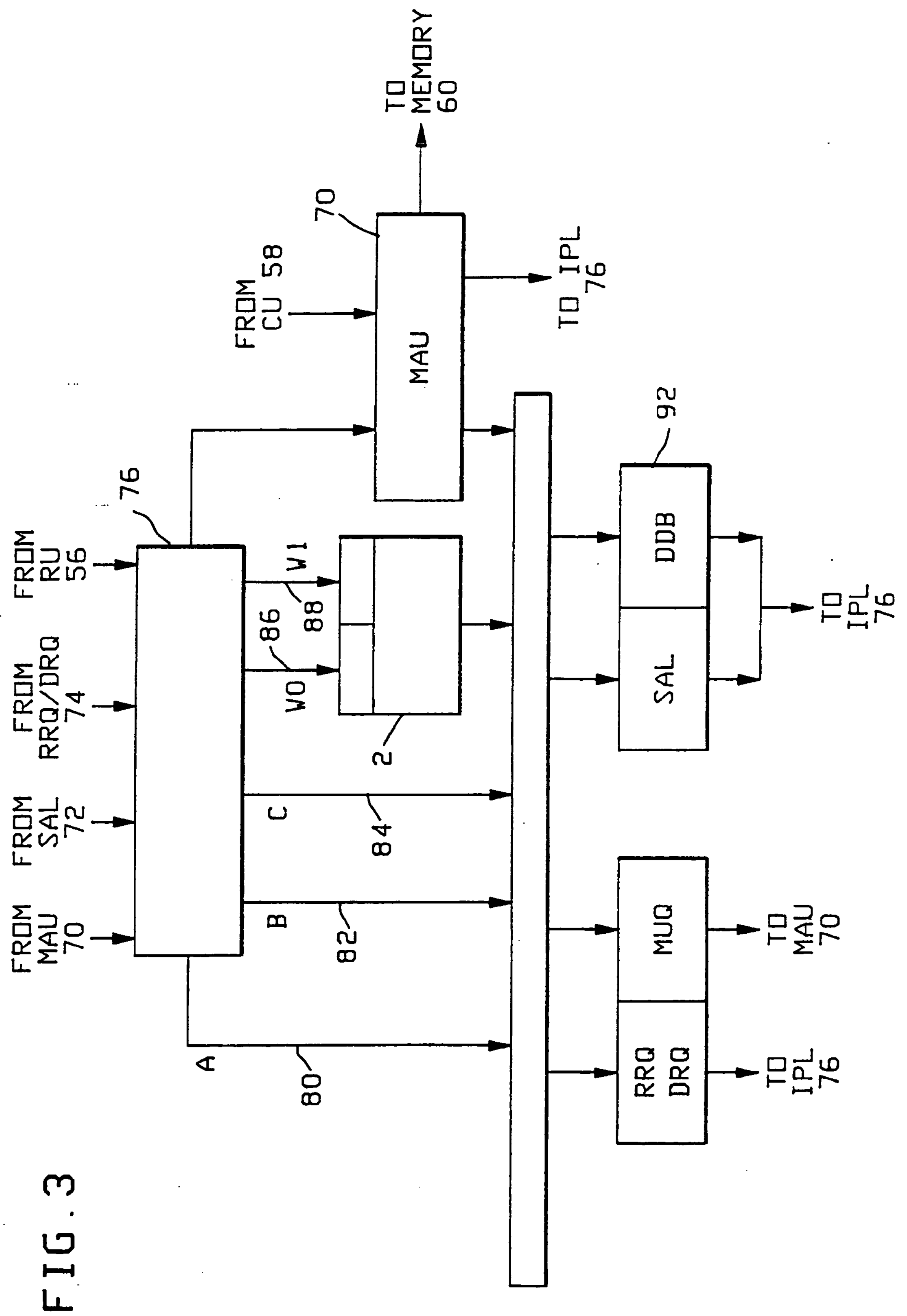

TO MEMORY 60
70
FROM CU 58
MAU
TO IPL 76
76
FROM RU 56
FROM RRQ/DRQ 74
FROM SAL 72
FROM MAU 70
W1
88
86
W0
2
C
84
B
82
A
80
92
DDB
SAL
TO IPL 76
MUQ
TO MAU 70
RRQ DRQ
TO IPL 76

FIG. 3